CSE 332: Data Structures and Parallelism

Exercises (Hashing)

Directions: *Submit your solutions on **Gradescope**. You must submit a pdf file.*

EX08. Hashibboleth (20 points)

(a) [12 Points] Valid/Invalid Hash Codes

For each of the following implementations of the `hashCode()` method, determine whether if the choice of hash code is valid. If the hash code is valid, point out why the hash code isn't optimal and how it might be improved. If the hash code is invalid, justify why it might not work. We simply consider an object's hash code to be valid if you can correctly insert, lookup and delete objects from a hash table without mistake.

1. For the `Double` class, you return -332 for the hash code if the number is less than 0, or 332 otherwise.
2. For the `String` class, you decide to just use Java's default implementation of hash codes, `Object.hashCode()`. Java typically uses the internal memory address of an object to determine its hash code.
3. For the `CircularArrayFIFOQueue` class, you return the current UNIX timestamp for the hash code. The UNIX timestamp is the number of seconds between January 1, 1970 and right now.

(b) [8 Points] Unlikely Scenarios

Imagine you are working with a map of type `HashMap<EX08String, EX08String>`. The `EX08String` class works almost exactly like Java's built-in `String` class, including how it implements the `hashCode()` method. The only difference is that Java's `String` class is immutable once initialized, whereas `EX08String` gives you a public method to modify the String afterwards.

For each of the following scenarios, determine whether the outcome is *Always*, *Maybe* or *Never*. Be sure to justify your answer.

1. You insert an item into the map. Then, you modify the key's characters. Will you be able to retrieve that item again?
2. You insert an item into the map. Then, you modify the value's characters. Will you be able to retrieve that item again?